VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Jiří Rýdel

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Rýdel**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: GIRITON Systems s.r.o.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Zdeněk Sawa, Ph.D.**

Konzultant bakalářské práce: Jan Gřeš, MSc.

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Rád bych na tomto místě poděkoval panu Ing. Janovi Gřešovi, MSc. za odbornou pomoc, příkladné vedení, ochotu a trpělivost v průběhu bakalářské praxe. Děkuji také vedoucímu bakalářské práce Ing. Zdeňkovi Sawovi, Ph.D.

Abstrakt

Cílem této bakalářské práce je popsat celkový průběh mé odborné praxe ve firmě GIRITON Systems s.r.o., která se zabývá vývojem informačních systémů pro řízení a monitorování především výrobních procesů. V rámci své stáže jsem ve firmě pracoval na pozici Java programátora a podílel se na vývoji informačního systému pro řízení výroby a sběru dat malých a středních podniků. Převážná část práce spočívala v tvorbě grafického uživatelského rozhraní systému.

Úvodní část této práce je věnována seznámení s firmou GIRITON Systems s.r.o., informačním systémem GIRITON a jsou zde popsány používané nástroje a technologie. Následující část obsahuje seznam jednotlivých úkolů a popis mých řešení. Závěrečná část obsahuje celkové zhodnocení praxe a výčet získaných znalostí.

Klíčová slova: Individuální odborná praxe, Java, Vaadin, Grafické uživatelské rozhraní, Jabber, Android, Integrační testy

Abstract

The purpose of this bachelor thesis is to describe an overall process of my professional practice in the GIRITON Systems s.r.o. company, that develops modern enterprise software. During the practice, I was involved in the development of information system for production management and data collection of small and medium-sized enterprises as a Java programmer.

The introductory part of this thesis is devoted to a brief characterization of company GIRITON Systems s.r.o., information system GIRITON and to a description of the development tools and technologies. The following section contains particular tasks and my solutions. The final part provides an overall assessment of my professional practice and a list of acquired knowledge.

Keywords: Individual professional practice, Java, Vaadin, Graphical User Interface, Jabber, Android, Integration tests

Seznam použitých zkratek a symbolů

HTML	–	HyperText Markup Language
SQL	–	Structured Query Language
POJO	–	Plain Old Java Object
HQL	–	Hibernate Query Language
POM	–	Project Object Model
XMPP	–	Extensible Messaging and Presence Protocol
CSS	–	Cascading Style Sheets
XML	–	Extensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V průběhu studia na Vysoké škole báňské - Technické univerzitě jsem během jednotlivých semestrů poznával nové technologie. V druhém semestru jsem se poprvé setkal s programovacím jazykem Java, který mě velmi zaujal. S jeho pomocí jsem začal vytvářet malé projekty a testovat jednotlivé funkcionality. Když mi byla nabídnuta možnost vypracovat bakalářskou práci formou odborné praxe, neváhal jsem a této šance ihned využil. Hlavním důvodem bylo především získání nových zkušeností a uplatnění těch v průběhu studia získaných. Při výběru jsem se soustředil převážně na ty firmy, které nabízely praxi na pozici právě Java programátora. Ze všech nabízených firem mě nejvíce zaujala firma GIRITON Systems s.r.o., která mi následně možnost vykonání bakalářské praxe nabídla.

V první části bakalářské práce nejprve představuji firmu GIRITON Systems s.r.o., její hlavní projekt a také mé pracovní zařazení. Následuje stručný popis používaných technologií a nástrojů. Další kapitola je zaměřená na tvorbu grafického uživatelského rozhraní informačního systému GIRITON. Tato část je rozdělená do čtyř podkapitol, ve kterých jsou popsána jednotlivá zadání úkolů a jejich následné řešení.

V následující sekci se věnuji implementaci Jabber serveru, který slouží pro komunikaci mezi uživateli v rámci informačního systému. Další dvě části jsou zaměřené na automatické odesílání e-mailů a tvorbu Android aplikace. Poslední část se věnuje testování grafického uživatelského rozhraní informačního systému pomocí integračních testů Selenium. Konkrétně se jedná o testy pro Vložení výrobního procesu v sekci 4.5.1 a Vymazání výrobního procesu v sekci 4.5.2. Oba tyto testy naleznete na straně 15.

V závěru práce jsou zhodnoceny celkové přínosy a vědomosti získané v průběhu praxe a také to, do jaké míry jsem využil znalosti získané na Vysoké škole báňské - Technické univerzitě v Ostravě.

2 Informace o praxi

2.1 GIRITON Systems s.r.o.

GIRITON Systems s.r.o. [1] je mladá rozvíjející se firma zabývající se tvorbou zakázkových informačních systémů. V současné době se firma zaměřuje obecně na tvorbu podnikových informačních systémů používajících mobilní terminály, tablety a chytré telefony pro automatický i ruční sběr dat a obousměrnou komunikaci. Mezi poskytované služby patří dále správa, sdílení, automatická synchronizace, zálohování a verzování dokumentů a jiných dat mezi počítači, tablety a chytrými telefony zákazníka. V neposlední řadě firma vytváří, dodává a provozuje zakázkové aplikace na podporu firemních procesů, a to jak klasické, tak cloudové aplikace přístupné odkudkoliv po internetu. Firma sídlí v Podnikatelském inkubátoru na půdě Vysoké školy báňské – Technické univerzity v Ostravě. Logo firmy můžete vidět na následujícím obrázku. (*Obrázek č.* 1)

Obrázek 1: Logo firmy

2.2 Informační systém GIRITON

GIRITON [2] je informační systém, sloužící k řízení a monitorování především výrobních procesů. Jednotlivá data jsou zaznamenávána pomocí mobilních terminálů, tabletů a dalších senzorů a následně pomocí bezdrátových technologií odeslána do informačního systému. Ve výsledku tak má uživatel aktuální informace o procesech, stavech skladů, jednotlivých zakázkách a dalších informace o aktuálním stavu výroby. Součástí produktu je volitelně i docházkový systém, díky kterému je možno tvořit mzdové podklady zaměstnanců.

2.3 Pracovní zařazení a náplň práce

V době mého příchodu se firma zabývala implementací webového informačního systému GIRITON. Tento systém již fungoval ve formě desktopové aplikace a jednotlivé funkcionality byly testovány ve společnosti na výrobu obuvi. Během prvních dvou dnů jsem byl důkladně seznámen s tímto systémem, používanými technologiemi a celým týmem. Prvním úkolem, který mi byl v rámci seznámení zadán, bylo prostudovat dokumentaci k samotnému frameworku a následně implementovat jednoduché zadání grafického

uživatelského rozhraní. S pomocí dokumentace jsem tento úkol splnil a získal tak základní přehled o jednotlivých grafických komponentách a jejich použití. Následně jsem byl zařazen na pozici Java programátora a byl mi přidělen další úkol, a to implementace tabulky typu strom pro editor kategorií, který je podrobně popsán v sekci 4.1.1 na straně 8. V průběhu praxe jsem práci prováděl převážně samostatně, u složitějších úkolů mi byl vždy někdo k dispozici a mohl jsem danou problematiku prodiskutovat a nechat si poradit s následnou implementací.

3 Používané technologie

3.1 Vaadin

Vaadin [3] je softwarový framework pro tvorbu webových aplikací. Kód je psán v jazyce Java a pomocí Google Web Toolkitu překládán do JavaScriptu, který je interpretován v internetovém prohlížeči. Obsahuje velkou sadu běžně používaných komponent pro tvorbu webových aplikací. Velikou výhodou Vaadinu je podpora pro nejpoužívanější internetové prohlížeče a programátor se tedy nemusí starat o ladění aplikace v různých prostředích. Spuštění aplikace na klientské straně nevyžaduje žádný dodatečný plugin a ani Javu. Stačí jen prohlížeč s podporou HTML a JavaScriptu.

3.2 Hibernate

Hibernate [4] je framework napsaný v jazyce Java, který pomocí objektově relačního mapování umožňuje namapovat doménový model na relační databázi. Hlavní funkcionalitou je tedy mapování Java objektů na entity v relační databázi a konverze Java datových typů na datové typy SQL. K tomu používá buď tzv. mapovací soubory, ve kterých je popsáno, jakým způsobem se mají data z objektu transformovat do databáze a naopak, nebo anotace. Pracuje s klasickými POJO objekty a poté, co jsou tyto objekty uloženy v databázi, se na ně lze dotazovat jazykem HQL, který je odvozen z SQL a je mu tedy velmi podobný.

3.3 Apache Maven

Apache Maven [5] je nástroj pro správu, řízení a automatizaci buildů aplikací. Základním principem fungování Mavenu je popsání projektu pomocí POM. Tento model popisuje softwarový projekt nejen z pohledu jeho zdrojového kódu, ale včetně závislostí na externích knihovnách, popisu procesu buildování a různých funkcí s tím spojených (jako je spouštění testů, sbírání informací o zdrojových kódech a podobně). Maven umožňuje rozdělit proces buildu do více fází. Tento přístup umožňuje nejen automaticky spouštět pluginy, ale také specifikovat fázi, ve které má Maven skončit.

3.4 HyperSQL databáze

HSQLDB [6] je relační databázový systém napsaný v jazyce Java. Používá čtyři hlavní typy tabulek pro ukládaní a čtení dat. Jedná se o malý, vícevláknový a transakční nástroj, který umožňuje operace nad tabulkami, a to jak v paměti, tak i přímo na disku.

4 Úkoly řešené v průběhu praxe

4.1 Tvorba grafického uživatelského rozhraní

Jak už bylo zmíněno, v době mého příchodu do firmy bylo hlavní prioritou přetvořit informační systém z desktopové aplikace na webovou. Tvorba grafického uživatelského rozhraní byla tedy hlavní náplní práce v průběhu praxe. K dispozici jsem měl původní aplikaci včetně zdrojových kódů.

4.1.1 Tabulka typu strom pro editor kategorií

Ve výrobním procesu spadá každý materiál do určité kategorie. Příkladem takové kategorie z testovacích dat je kategorie *materiál/látky/pánská/zimní*.

Mým prvním úkolem bylo vytvořit tabulku stromové hierarchie, která bude umět tyto kategorie správně zobrazit a pracovat s nimi. K dispozici byla databáze firmy na výrobu obuvi, která obsahovala testovací data pro následné ověření funkčnosti. Každý řádek tabulky měl kromě názvu kategorie obsahovat i checkbox pro její vybrání.

Pro vypracování jsem zvolil komponentu *TreeTable*. Jedná se o tabulku, která je potomkem základní *Table* a implementuje rozhraní *Hierarchical*, takže umožňuje data zobrazit ve formě stromu s jedním hlavním kořenem. Důležité bylo také vytvořit mapování kategorie-checkbox a také celkové hierarchické řazení kategorií.

Tabulka obsahuje 3 základní sloupce:

- CheckBox
- Kategorie předek
- Kategorie potomek

Každý řádek tabulky tedy obsahuje checkbox, danou kategorii a jejího předka. Viditelnými prvky v tabulce jsou pouze checkboxy, u kterých je titulek nastaven na název kategorie-potomek. O vytvoření této tabulky se stará metoda *createContainer*, které v parametru předávám kolekci kategorií, získanou z databáze. V první fázi se tabulka naplní daty a pro každý řádek se vytvoří nový checkbox, který se následně i s kategorií potomek uloží do mapy. Objekty se ukládají do mapy proto, aby se v budoucnu při vybrání některé z kategorií mohl zaškrtnout checkbox na stejném řádku. V druhé fázi se tabulce nastavuje závislost předek-potomek.

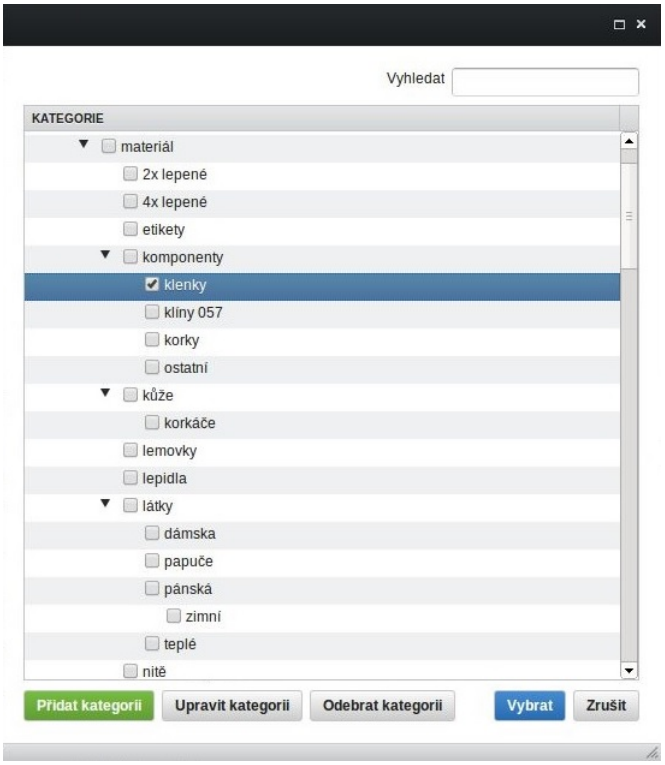
Třída dále obsahuje metodu, která v tabulce zvýrazní řádek dané kategorie předané v parametru metody a posune se tak, aby byl tento prvek přibližně v polovině zobrazené části tabulky. Dále jsem implementoval listener pro checkbox, který při zaškrtnutí zvýrazní daný řádek a nastaví zvolenou kategorii.

4.1.2 Editor kategorií

Po dokončení implementace tabulky pro práci s kategoriemi byl můj další úkol vytvořit grafické rozhraní a zařadit tak práci s kategoriemi do informačního systému. Třída grafického uživatelského rozhraní se skládá ze dvou částí.

- Komponenta obsahující celkové zařazení kategorie a možnost výběru.
- Okno pro zobrazení, výběr, úpravu nebo vytvoření nové kategorie. (*Obrázek č.* 2)

V první části jsem vytvořil komponentu pro zobrazení stromové cesty zvolené kategorie až po kořenového předka. Při výběru materiálu se v textovém poli zobrazí tato cesta kategorie. Po kliknutí na tlačítko *Vybrat* se zobrazí nové okno s tabulkou, ve které jsou zobrazeny všechny kategorie. Je zde možné kategorii vybrat, vytvořit, smazat nebo upravit. Pokud je při vytváření nové kategorie označena některá z kategorií v tabulce, nová kategorie se vytvoří jako její potomek. Jednotlivých kategorií je mnoho, a proto je možnost kategorii i vyhledat pomocí názvu. V tabulce se při psaní filtrují jenom ty kategorie, které obsahují daný řetězec ve svém názvu.



Obrázek 2: Okno pro zobrazení kategorií

4.1.3 Panel pro odesílání zpráv a příloh

V pořadí třetím úkolem byla implementace komponenty pro odesílání zpráv a příloh. Mnou vytvořená třída je potomkem vertikálního layoutu, který vykresluje jednotlivé dílčí komponenty vertikálně pod sebou v pořadí, v jakém do něho byly vloženy. Obsahuje komponentu pro psaní textu, tlačítka pro odeslání zprávy a připojení přílohy a layout pro zobrazení nahraných příloh.

Konstruktor obsahuje dva parametry. První z nich je samotný listener na tlačítko pro odesílání, jelikož se tato komponenta používá na více místech a každé z nich tuto funkcionalitu využívá různě. Druhým parametrem je proměnná, která definuje, zda je možné odesílat zprávy pomocí klávesy Enter. Při stisknutí této klávesy se programově vyvolá stisknutí tlačítka *Odeslat*.

Další funkcionalitou komponenty pro psaní textu je možnost *DragAndDrop* souborů, které chceme připojit. Nemusíme tedy vybírat soubory v prohlížeči souborů, ale stačí pouze soubory do této komponenty přetáhnout. Tato komponenta navíc umožňuje kontrolovat přílohu přetaženého souboru.

Dále jsem implementoval dvě vnitřní třídy. První z nich, *ChatMessageAttachment*, v sobě obsahuje název souboru, příponu a samotný soubor ve formě pole bajtů. Druhá třída, *ChatAttachmentLayout* vytváří layout s obrázkem na základě toho, zda je soubor připraven k odeslání nebo přijat.

4.1.4 Agenda pro docházku

Jedním z časově náročnějších úkolů bylo vytvořit celkovou agendu pro kontrolu docházky a přehledně tak zobrazit informace o docházce zaměstnanců za určité období.

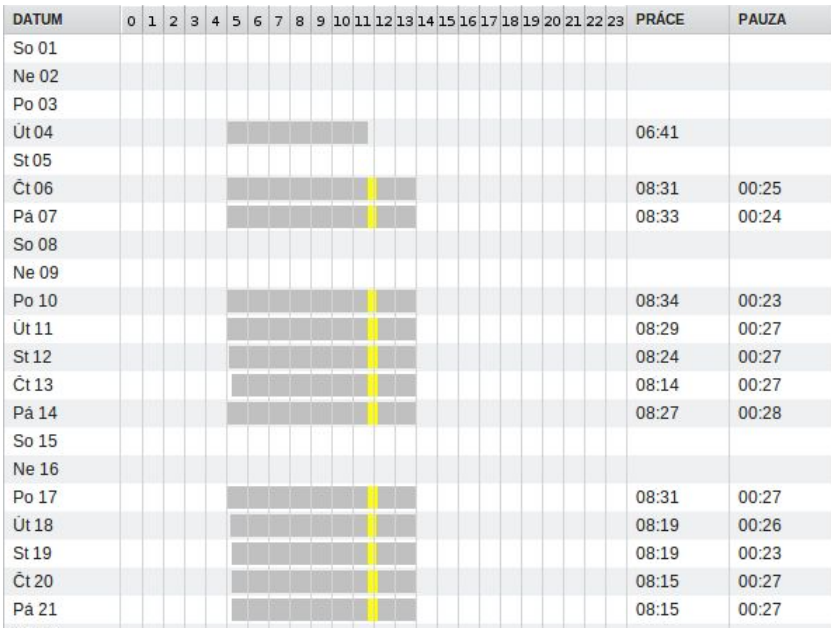
Jako první jsem implementoval vlastní tabulku, která se oproti základní *Table* skládá ze dvou tabulek, přičemž jedna z nich slouží jako hlavička. Pomocí CSS stylů jsem docílil identické podoby a tabulky vypadají stejně. Tuto tabulku bylo nutné vytvořit, jelikož jedna z podmínek v zadání úkolu byla možnost budoucího vkládání komponent do hlavičky tabulky. Tuto funkcionalitu základní *Table* neumožňuje.

Finální layout se skládá ze dvou tabulek a komponenty pro výběr data. V levé tabulce jsou vypsáni všichni zaměstnanci, kteří jsou pro zvolené období ve firmě zaměstnaní. Pravá tabulka vykresluje docházku vybraného zaměstnance. Nad tabulkou zaměstnanců se nachází komponenta pro výběr časového úseku, pro který chceme údaje zobrazit. Na tuto komponentu je zaregistrovaný listener, který při výběru data nejprve zkontroluje, zda je vybrána i osoba a následně aktualizuje tabulku s docházkou.

Docházková tabulka se skládá z pěti sloupců. (*Obrázek č.* 3)

- Datum - zde jsou vypsány všechny dny pro zvolený měsíc.
- Hodiny - zde jsou vypsány všechny hodiny pro daný den.
- Práce - zde je vypsán celkový počet odpracovaných hodin.
- Pauza - zde je vypsán celkový počet hodin pauzy.

Do druhého sloupce docházkové tabulky se vkládá vytvořený obrázek, ve kterém jsou různými barvami znázorněny pracovní bloky. Obrázky vykresluje třída *DayActivityImage*, která implementuje rozhraní *StreamSource*. Vytvořený objekt této třídy v konstruktoru převezme vybraný den a následně projde všechny pracovní záznamy pro tento den. Každý pracovní záznam obsahuje čas začátku, čas konce a o jaký typ aktivity se jedná. Jelikož má tento sloupec v tabulce danou pevnou velikost v pixelech, mohl jsem implementovat metodu, která pomocí procent vypočítá z časového intervalu aktivity úvodní a konečný pixel. Mezi těmito pixely jsem následně vykreslil obdélník v barvě dané aktivity.



Obrázek 3: Tabulka docházky

4.2 Jabber/XMPP Server

XMPP [7], dříve známý jako Jabber, je komunikační protokol pro zasílání XML zpráv a zjišťování stavu. Síť XMPP je založena na architektuře klient-server (klienti zpravidla nekomunikují přímo) a je decentralizována stejně jako e-mail. Vlastní server může zřídit kdokoliv, přičemž bude moci komunikovat s uživateli na jiných serverech. Základním elementem protokolu XMPP jsou *Stanzy*.

Úkolem bylo implementovat vlastní XMPP server a vytvořit tak chat v rámci informačního systému. Pro tento úkol jsem se rozhodl využít knihovnu Apache Vysper [8]. V první řadě bylo potřeba upravit třídu *MyMemoryRosterManager*, který v běhu spravuje jednotlivé seznamy uživatelů, a implementovat metodu pro vzájemné přidávání uživatelů do kontakt listů. V dalším kroku jsem vytvořil třídu *MyMutableRoster*, která v sobě mapuje jednotlivé uživatele a jejich seznam kontaktů. Při implementaci třídy *MyXMPP-Server* bylo nutné implementovat listener, který bude se *Stanzou* dále pracovat podle jejího typu.

V našem případě rozlišujeme typy dva.

- Zpráva
- Změna stavu

Podstatná práce probíhala ve třídě *MyServer*. Tato třída obsahuje základní metody pro start a stop serveru. Důležitou metodou je také metoda *deliverMessage*, která v parametru přijímá objekt ChatMessage. Objekt je v rámci metody zpracován a je vytvořena Stanza, kterou následně server odešle.

Při startu webové aplikace se automaticky zapne i server. Aby mohli uživatelé mezi sebou komunikovat, bylo nutné vytvořit jejich účty a přidat do kontaktního listu každého uživatele všechny ostatní. Pro vytvoření uživatele na serveru jsem implementoval metodu *addUser*, kterou můžete vidět v následujícím výpisu zdrojového kódu. (*Výpis č.* 1) Tato metoda nejprve zkontroluje, zda už jiný uživatel se stejným jménem neexistuje, nebo zda už nebyl přidán. Po přidání uživatele je nutné do jeho kontakt listu vložit všechny ostatní uživatele.

Celková práce nakonec probíhala v rámci pěti tříd a konečný počet řádků byl 642.

```
public void addUser(String userName, String password, Actor a, Object tenantId) {
        try {
            if (!takenUserNames.add(userName)) {
                throw new ProgramException("Uzivatelske jmeno je jiz obsazeno");
            }
            if (actorIdToActorCache.put(a.getId(), a) != null) {
                throw new ProgramException("Tento uzivatel jiz byl do Jabber serveru pridan");
            }

            Entity user = EntityImpl.parseUnchecked(userName);
            actorToJidCache.put(user.getBareJID(), a);
            actorToTenantIdCache.put(a, tenantId);
            accountManagement.addUser(user.getBareJID(), password);

            providerRegistry.getMyMemoryRosterManager().addNewRosterInternal(user);

            Map<Entity, MyMutableRoster> tempMap = providerRegistry.
                    getMyMemoryRosterManager().getRosterMap();
            for (Entity e : tempMap.keySet()) {
                Object eTenantId = actorToTenantIdCache.get(actorToJidCache.get(e.getBareJID())
                        );
                if (e == user || !eTenantId.equals(tenantId)) {
                    continue;
                }
                providerRegistry.getMyMemoryRosterManager().addFriend(user, e);
            }
        } catch (Throwable ex) {
            log.error("Chyba pri pridavani uzivatele " + userName, ex);
        }
    }
```

Výpis 1: Metoda pro přidání uživatele do Jabber serveru

4.3 Tvorba Android aplikace

Android [9] je v dnešní době nejrozšířenějším operační systém pro mobilní zařízení. Ve firmě GIRITON Systems s.r.o. se využívají mobilní zařízení s tímto operačním systémem pro sběr jednotlivých informací, které jsou následně odesílány do informačního systému.

Mým dalším úkolem bylo tedy vytvořit základ nové android aplikace včetně otestování správné funkčnosti a připravit tak základ pro budoucí vývoj. Aplikace má běžet v pozadí a upozorňovat na sebe notifikační ikonkou. Po kliknutí na tuto ikonku se otevře uživatelské rozhraní, ve kterém je možné odeslat SMS zprávu na zadané číslo. Další podmínkou je automatické zapnutí aplikace při startu zařízení.

S vývojem na platformě android jsem se do té doby nesetkal, proto bylo důležité nejprve pročíst dokumentaci. Následně mi byl přidělen jeden z kolegů, který mi vysvětlil základní principy při tvorbě aplikace.

Po vytvoření projektu bylo nutné nejprve vytvořit základní uživatelské rozhraní, které jsem definoval v XML souboru a vytvořil jsem základní aktivitu pro odesílání zpráv.

Pro umožnění funkce odesílání zpráv a autostartu aplikace jsem do souboru *AndroidManifest.xml* přidal následující oprávnění.

- `<uses-permission android:name="android.permission.SEND_SMS"/>`
- `<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>`

Jak už bylo zmíněno, aplikace měla běžet v pozadí, a proto jsem vytvořil *Service*, který na svoji přítomnost upozorňuje notifikační ikonkou. Tento servis na rozdíl od aktivity běží v pozadí a neposkytuje uživatelské rozhraní. Dále jsem implementoval třídu *StartupReceiver*, která po kompletním startu zařízení servis spustí.

Nakonec jsem tuto aplikaci několikrát otestoval a předal kolegům pro další práci.

4.4 Automatické odesílání e-mailů

Pokud je prázdný sklad, určitá objednávka je hotová a nebo nastane jiná událost, o které chce být zákazník informován, má možnost nastavit si informativní e-maily. Mým úkolem bylo implementovat třídu, která bude obsahovat metody na odeslání a přijetí těchto e-mailů.

Pro implementaci jsem použil základní knihovny *JavaMail* [10]. Vytvořil jsem testovací e-mail a potřebné informace zapsal do souboru *Server.xml*.

Tyto informace mohu následně získat z aktuálně běžícího prostředí. Po nastavení textu a předmětu se zpráva odešle všem vybraným adresám.

4.5 Tvroba integračních testů

Důležitou součástí při vývoji informačního systému je jeho testování. V informačním systému GIRITON se pro testování grafického uživatelského rozhraní používají Selenium testy [11].

Jedná se o komplexní sadou nástrojů pro automatizované testování webových aplikací. Tento framework umožňuje tvorbu a spouštění skriptů, které popisují chování virtuálního uživatele na webu a kontroly, která ověřuje, zda se testovaný web chová podle očekávání.

Mým úkolem bylo naprogramovat dva testy pro kontrolu funkčnosti grafického uživatelského rozhraní při práci s výrobními procesy. První test se v grafickém rozhraní automaticky přihlásí, otevře požadovanou agendu, vyplní patřičné údaje, které jsou potřeba pro vložení, a následně záznam uloží. Druhý test některý ze záznamů vymaže. Po obou testech probíhá kontrola databáze a kontroluje se, zda změny proběhly v pořádku.

Všem grafickým komponentám, se kterými testy pracují, bylo potřeba nastavit identifikátor. V testu jsem pak tyto komponenty získal vyhledáním v Document Object Modelu pomocí nastaveného identifikátoru.

4.5.1 Vložení výrobního procesu

V prvním testu je nejprve potřeba zjistit, kolik výrobních procesů již databáze obsahuje. Celkový počet záznamů je nutné znát pro pozdější kontrolu databáze. Dále je nezbytné do databáze vložit záznamy potřebné pro uložení výrobního procesu. Pokud obě operace nad databází proběhnou úspěšně, přichází na řadu samotný test.

Nejdříve se spustí okno prohlížeče a test se přihlásí do aplikace. Po přihlášení test zvolí agendu *Správa* a klikne na položku *Výrobní procesy*. Do jednotlivých kolonek vyplní zadané údaje. Poté, co jsou potřebné údaje vyplněny, přepne test na záložku *Model* a v tabulce klikne na tlačítko *Přidat*. Pomocí metody *selectTableRowWithCTRL* (*Výpis č.* 3 test označí dva záznamy a stiskne tlačítko *OK*. V posledním kroku test klikne na tlačítko *Uložit* a potvrdí dialog upozorňující na uložení dat.

Finální kontrola databáze kontroluje, zda je aktuální počet záznamů v databázi o jeden větší, než byl při kontrole před začátkem testu.

4.5.2 Vymazání výrobního procesu

V druhém testu nejprve opět proběhne kontrola databáze. Kontroluje se, zda je počet výrobních procesů větší než nula a aktuální počet záznamů se uloží do proměnné. Pokud je kontrola úspěšná, přijde na řadu samotné vymazání výrobního procesu. Nejprve se test přihlásí a otevře agendu *Výrobní procesy*. Následně označí záznam v tabulce a klikne na tlačítko vymazat. Po odkliknutí dialogu o změně údajů se výrobní příkaz vymaže. Nakonec proběhne poslední kontrola databáze, která kontroluje, zda je aktuální počet výrobních procesů v databázi o jeden menší než počet před proběhnutím testu. Test pro vymazání výrobního příkazu můžete vidět na následujícím výpisu. (*Výpis č.* 2)

```
@Test
    public void test2removeWFSuccess() throws Exception {
        List<WfModel> ents = rmi.executeJpqlQuery("SELECT ent FROM " + WfModel.class.
            getSimpleName() + " ent ", null, Boolean.FALSE);
        int countBefore = ents.size();
        Assert.assertTrue("Table is empty!", countBefore > 0);

        LoginUtils.completeLogin(driver, wait);
        TopMenuBarUtils.selectFromPopUpMenu(MENU_ITEM_NAME, popUpItemIndex, driver,
            wait);

        WebElement root = wait.until(ExpectedConditions.presenceOfElementLocated(By.id(
            ClientPluginPanelStandardVaadin.TABLE_ID)));
        WebElement table = root.findElement(By.className(TABLE_CLASS_NAME));
        WebElement row = table.findElement(By.xpath("./tbody/tr[1]"));
        row.click();

        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(
            ClientPluginPanelStandardVaadin.DELETE_BTN_ID))).click();
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(EntryDeleterVaadinImpl.
            FIRST_DELETE_DIALOG_CONFIRM_BTN_ID))).click();
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.id(EntryDeleterVaadinImpl.
            SECOND_DELETE_DIALOG_CONFIRM_BTN_ID))).click();

        Thread.sleep(PAUSE);

        ents = rmi.executeJpqlQuery("SELECT ent FROM " + WfModel.class.getSimpleName() +
            " ent ", null, Boolean.FALSE);
        Assert.assertEquals("Wrong number of records!", countBefore - 1, ents.size());
    }
```

Výpis 2: Test pro úspěšné odebrání výrobního procesu

```
private void selectTableRowWithCTRL(String window, String tableName, int row) {
        Actions shiftClick = new Actions(driver);
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.className(window)));
        String tableRow = "//table[@class='" + tableName + "']/tbody/tr[" + row + "]";
        WebElement rowElement = driver.findElement(By.xpath(tableRow));
        shiftClick.keyDown(Keys.CONTROL).click(rowElement).keyUp(Keys.CONTROL).perform
            ();
}
```

Výpis 3: Metoda pro vybrání více řádků

5 Závěr

5.1 Dosažené výsledky a jejich hodnocení

Absolvování odborné praxe je pro mě ve všech směrech velmi cenná zkušenost. Měl jsem možnost pracovat na reálném projektu a využít tak znalosti nabyté v průběhu studia. Naučil jsem se spolupracovat s týmem lidí, kteří pracují na stejném projektu. Získal jsem důležitý přehled o aktuálně používaných technologiích a jejich potenciálu využití. A v neposlední řadě jsem získal bohaté zkušenosti nejen v programování, ale i v oblasti používání mobilních zařízení pro sběr dat.

5.2 Uplatněné teoretické a praktické znalosti

Při absolvování praxe jsem nejvíce využil znalosti z předmětu *Základy programování I*, jelikož veškeré programování v průběhu mé praxe probíhalo v jazyce Java. Při práci s databázemi jsem měl velmi dobrý základ z předmětů *Úvod do databázových systémů* a *Databázové a informační systémy*. Dále jsem pak velmi ocenil nabyté znalosti a zkušenosti z předmětu *Uživatelská rozhraní* při práci na grafickém uživatelském rozhraní.

5.3 Scházející znalosti a dovednosti

Mezi scházející znalosti bych zařadil všeobecný přehled o knihovnách jazyka Java a jejich využití, které obsahují mnoho využitelných funkcí a v mnoha případech je to velké usnadnění práce. Dále pak pouze teoretické znalosti v dnešní době velmi často používaných frameworků jako je Hibernate nebo Vaadin. A také znalost omezeného množství různých databázových systémů. S databází HSQL jsem se setkal poprvé a velmi mě překvapila její jednoduchost a síla. Tuto databázi jsem následně využil i v semestrálním projektu do předmětu *Java technologie*.

6 Reference

[1] GIRITON Systems s.r.o. [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://giriton.cz/o-nas

[2] Informační systém GIRITON [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://giriton.cz

[3] Vaadin [online]. 2014 [cit. 2014-04-24].
Dostupné z: https://vaadin.com/

[4] Hibernate [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://hibernate.org/

[5] Apache Maven [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://maven.apache.org/

[6] HyperSQL Database [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://hsqldb.org/

[7] Extensible Messaging and Presence Protocol [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://xmpp.org/

[8] Apache Vysper [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://mina.apache.org/vysper-project/

[9] Android [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://www.android.com/

[10] JavaMail [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://www.oracle.com/technetwork/java/javamail/index.html

[11] Selenium [online]. 2014 [cit. 2014-04-24].
Dostupné z: http://docs.seleniumhq.org/